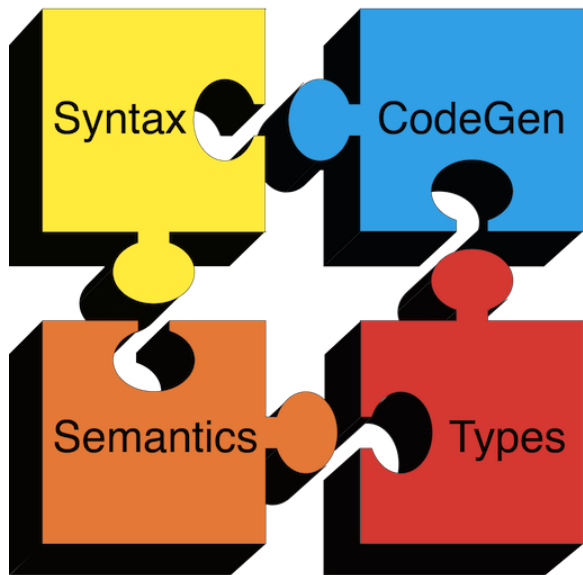




Compilers

Java Exceptions

- Deep in a section of code, you encounter an unexpected error
 - Out of memory
 - A list that is supposed to be sorted is not
 - etc.
- What do you do?

- Add a new type (class) of *exceptions*

- Add new forms

try { something } **catch**(x) { cleanup }

throw exception

```
class Foo {  
    public static void main(String[] args) {  
        try { X(); } catch (Exception e) {  
            System.out.println("Error!") } }  
  
    public void X() throws MyException {  
        throw new MyException();  
    }  
}
```

$T(v)$ = an exception that has been thrown with value v

v = an ordinary value (an object)

$$\frac{E \quad e_1 : v_1}{E \quad \text{try}\{e_1\} \text{ catch}(x) \{e_2\} : v_1}$$

$$\frac{\begin{array}{c} E \quad e_1 : T(v_1) \\ E[x \leftarrow v_1] \quad e_2 : v_2 \end{array}}{E \quad \text{try}\{e_1\} \text{ catch}(x) \{e_2\} : v_2}$$

$$\frac{E \quad e : v}{E \quad \text{throw } e : T(v)}$$

$$\frac{E \quad e_1 : T(v_1)}{E \quad e_1 + e_2 : T(v_1)}$$

- When we encounter a **try**
 - Mark current location in the stack
- When we **throw** an exception
 - Unwind the stack to the first **try**
 - Execute corresponding **catch**
- More complex techniques reduce the cost of **try** and **throw**

What happens to an uncaught exception thrown during object finalization?

- Methods must declare types of exceptions they may raise

`public void X() throws MyException`

- Checked at compile time
 - Some exceptions need not be part of the method signature
 - e.g., dereferencing null
-
- Other mundane type rules
 - `throw` must be applied to an object of type `Exception`